

Rapport de Projet :

Le membres du groupe :

- Thomas ROUX
- Moussa TRAORE
- Ryan MAMBOU DJEMTCHEMEU
- Abdul Kodir Adewale MOUNIROU

Sommaire :

1. Introduction
 - 1.1. Présentation du Projet
 - 1.2. Objectif du Projet
2. Modèle Conceptuel des Données (MCD)
 - 2.1. Description
 - 2.2. Diagramme ER
 - 2.3. Diagramme MCD :
3. Traitements Effectués
 - 3.1. Chargement des Données dans Azure Blob Storage
 - 3.2. Traitement des Données avec Databricks
 - 3.3. Création des Tables dans Databricks
 - 3.4. Transformation des Données
4. Visualisations dans Databricks
5. Repository de Scripts
6. Conclusion

1. Introduction

1.1 Présentation du Projet

Ce projet vise à mettre en place un Data Warehouse pour analyser les impacts des catastrophes naturelles. Les données proviennent du fichier source **public-emdat**, contenant des informations sur divers types de catastrophes à travers le monde, leurs impacts financiers, et leurs effets sur les populations.

1.2 Objectif du Projet

L'objectif est de structurer et de transformer les données brutes pour permettre des analyses approfondies, notamment :

- Déterminer les coûts financiers par année.
- Évaluer les pertes humaines par pays.
- Identifier les types de catastrophes les plus coûteux.

2. Modèle Conceptuel des Données (MCD)

2.1 Description

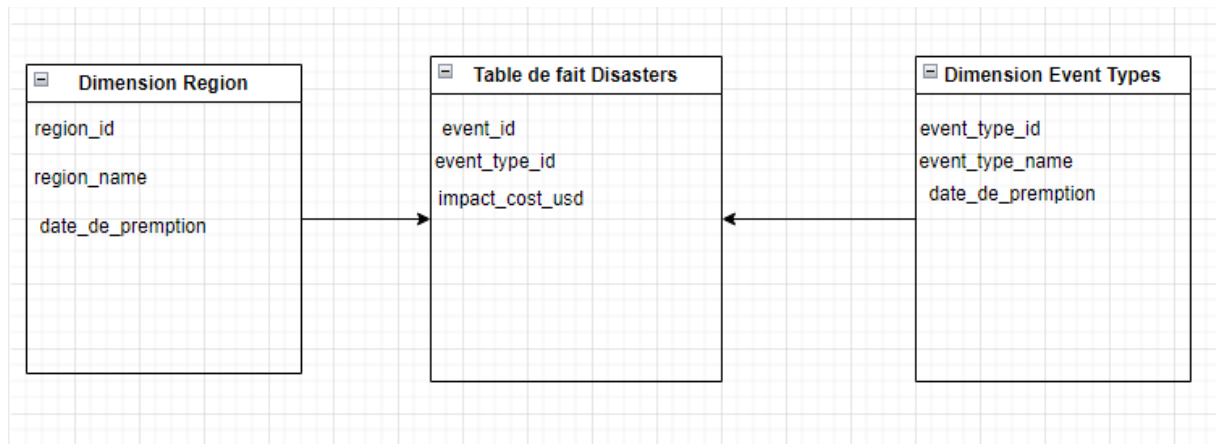
Le Data Warehouse est structuré en trois zones :

- **ds-bronze** : Données brutes chargées directement depuis le fichier source.
- **ds-silver** : Données nettoyées et enrichies avec des clés et des relations de base.
- **ds-gold** : Données optimisées pour l'analyse, avec une table de faits et des tables de dimensions.

2.2 Diagramme ER

- **Tables de Dimensions :**
 - **dim_regions** : ID de la région et nom de la région.
 - **dim_event_types** : ID de type d'évènement et son nom.
- **Table de Faits :**
 - **fact_disasters** : Contient les mesures (événements, coûts, victimes) et les clés étrangères vers les dimensions.

Diagramme MCD :



3. Traitements Effectués

3.1 Chargement des Données dans Azure Blob Storage

- Création des dossiers dans Azure Blob Storage :
 - /ds-bronze/emdat/current
 - /ds-silver/emdat/current
 - /ds-gold/emdat/current
- Commande pour charger le fichier public-emdat.parquet dans le dossier /ds-bronze/emdat/current :

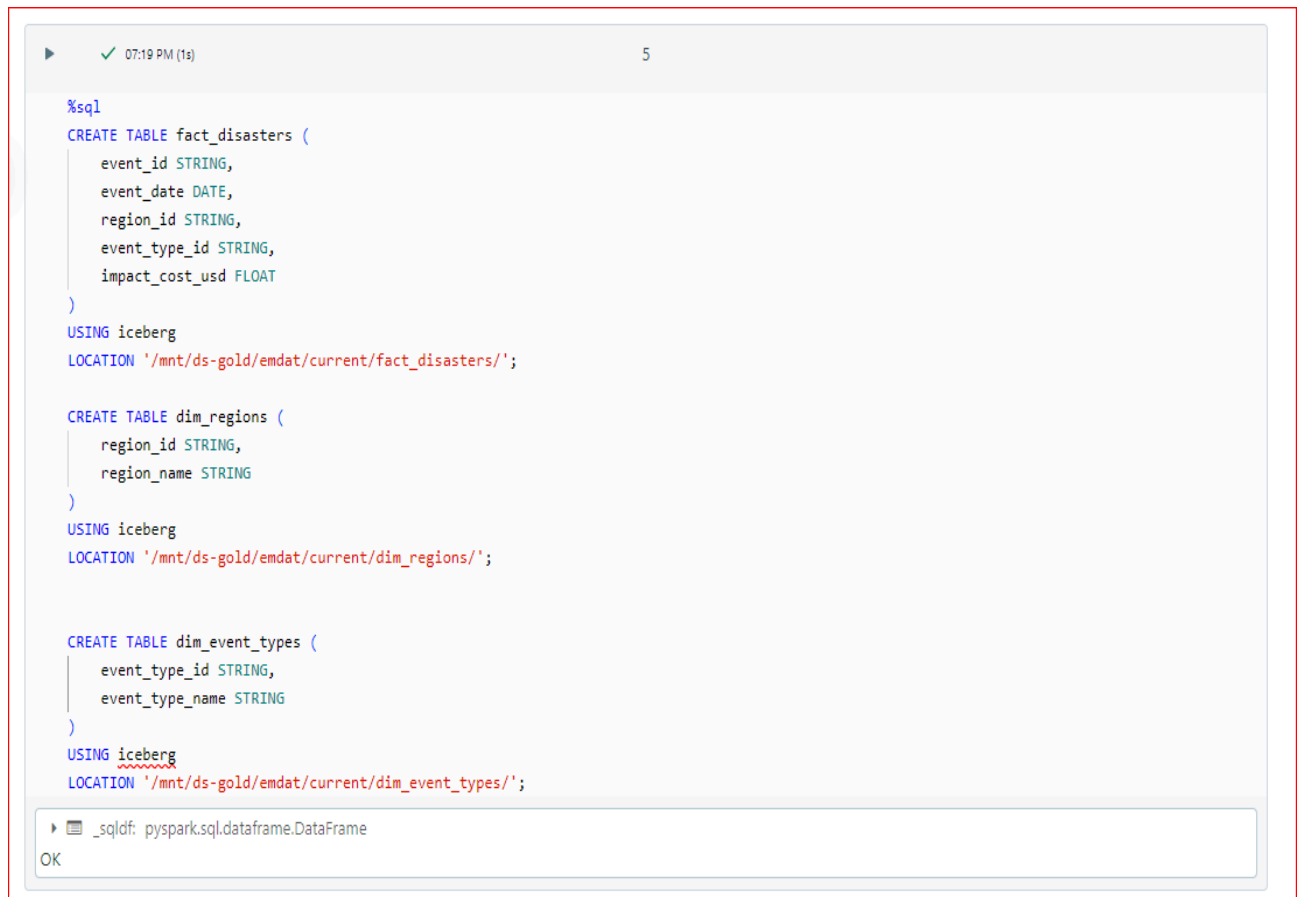
3.2 Traitement des Données avec Databricks

3.2.1 Création des Tables dans Databricks

- Définir la configuration Spark :

```
spark.conf.set("spark.sql.catalog.my_cataloggold", "org.apache.iceberg.spark.SparkCatalog")
spark.conf.set("spark.sql.catalog.my_cataloggold.type", "hadoop")
spark.conf.set("spark.sql.catalog.my_cataloggold.warehouse", "/mnt/ds-gold/emdat/current")
```

- **Tables créées :**




The screenshot displays a Databricks SQL interface. At the top, a status bar shows a green checkmark, the time '07:19 PM (1s)', and the page number '5'. The main area contains three SQL statements for creating tables using the Iceberg storage format. The first statement creates 'fact_disasters' with columns for event_id, event_date, region_id, event_type_id, and impact_cost_usd. The second creates 'dim_regions' with region_id and region_name. The third creates 'dim_event_types' with event_type_id and event_type_name. All tables are located in the '/mnt/ds-gold/emdat/current/' directory. At the bottom, a console output shows the execution of the first statement, resulting in a DataFrame named '_sqldf' of type 'pyspark.sql.dataframe.DataFrame', followed by an 'OK' status.

```
%sql
CREATE TABLE fact_disasters (
  event_id STRING,
  event_date DATE,
  region_id STRING,
  event_type_id STRING,
  impact_cost_usd FLOAT
)
USING iceberg
LOCATION '/mnt/ds-gold/emdat/current/fact_disasters/';

CREATE TABLE dim_regions (
  region_id STRING,
  region_name STRING
)
USING iceberg
LOCATION '/mnt/ds-gold/emdat/current/dim_regions/';

CREATE TABLE dim_event_types (
  event_type_id STRING,
  event_type_name STRING
)
USING iceberg
LOCATION '/mnt/ds-gold/emdat/current/dim_event_types/';
```

▶  _sqldf: pyspark.sql.dataframe.DataFrame
OK

3.2.2 Transformation des Données

- Code PySpark :

```
▶ 07:19 PM (56s) 10

from pyspark.sql import functions as F

dim_regions = silver_emdat.select("region").distinct() \
    .withColumnRenamed("region", "region_name") \
    .withColumn("region_id", F.monotonically_increasing_id()) # Génération d'un ID unique

dim_regions.write.format("iceberg").mode("overwrite").saveAsTable("my_cataloggold.dim_regions")

▶ (1) Spark Jobs
dim_regions: pyspark.sql.dataframe.DataFrame = [region_name: string, region_id: long]
```

```
▶ 07:20 PM (46s) 11

dim_event_types = silver_emdat.select("disaster_type").distinct() \
    .withColumnRenamed("disaster_type", "event_type_name") \
    .withColumn("event_type_id", F.monotonically_increasing_id()) # Génération d'un ID unique

dim_event_types.write.format("iceberg").mode("overwrite").saveAsTable("my_cataloggold.dim_event_types")

▶ (1) Spark Jobs
dim_event_types: pyspark.sql.dataframe.DataFrame = [event_type_name: string, event_type_id: long]
```

```
▶ 07:21 PM (1m) 12

fact_disasters = silver_emdat.join(dim_regions, silver_emdat["region"] == dim_regions["region_name"], "inner") \
    .join(dim_event_types, silver_emdat["disaster_type"] == dim_event_types["event_type_name"], "inner") \
    .select(
        F.monotonically_increasing_id().alias("event_id"),
        "event_date",
        dim_regions["region_id"],
        dim_event_types["event_type_id"],
        silver_emdat["total_damage_usd"].alias("impact_cost_usd")
    )

fact_disasters.write.format("iceberg").mode("overwrite").saveAsTable("my_cataloggold.fact_disasters")

▶ (3) Spark Jobs
fact_disasters: pyspark.sql.dataframe.DataFrame = [event_id: long, event_date: date ... 3 more fields]
```

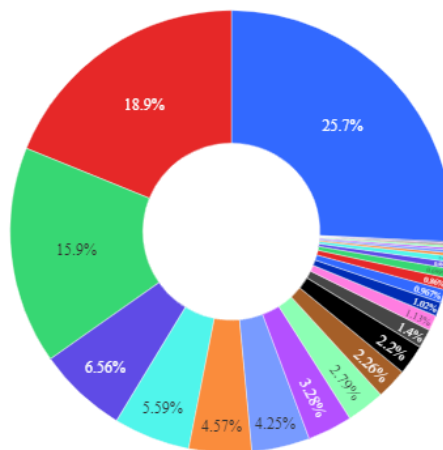
4. Visualisations dans Databricks

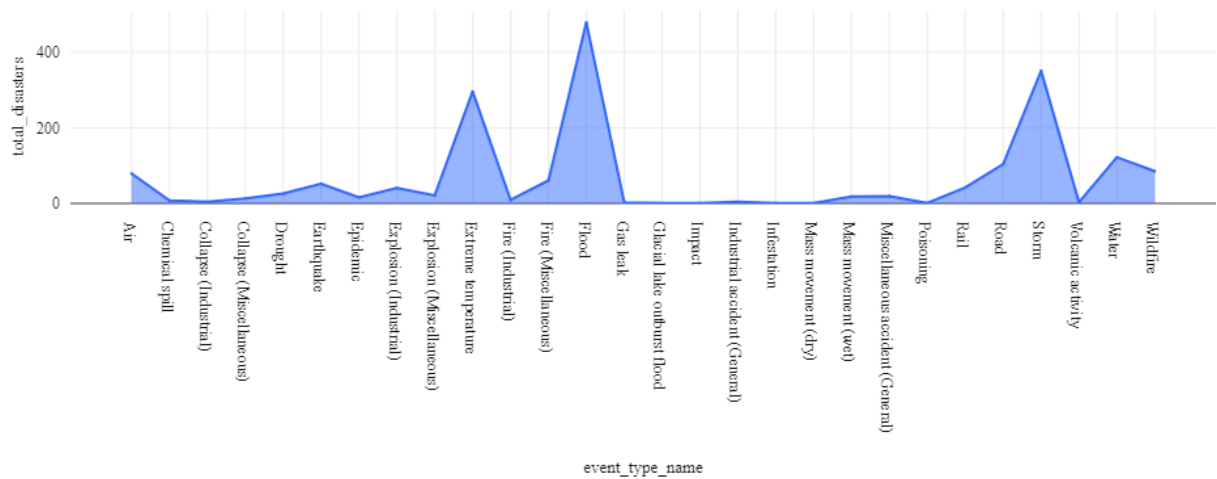
- Total des catastrophes par type :

```
%sql
SELECT
  det.event_type_name,
  COUNT(fd.event_id) AS total_disasters
FROM
  my_cataloggold.fact_disasters fd
JOIN
  my_cataloggold.dim_event_types det
ON
  fd.event_type_id = det.event_type_id
GROUP BY
  det.event_type_name
ORDER BY
  total_disasters DESC;
```

► (2) Spark Jobs

► `_sqldf: pyspark.sql.dataframe.DataFrame = [event_type_name: string, total_disasters: long]`





• Coût total des dommages par an :

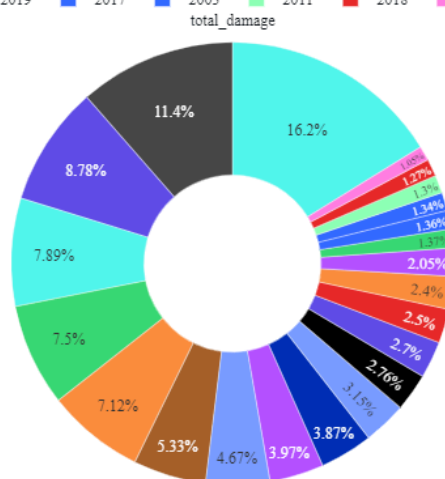
```

%sql
SELECT
  YEAR(fd.event_date) AS event_year,
  SUM(fd.impact_cost_usd) AS total_damage
FROM
  my_cataloggold.fact_disasters fd
GROUP BY
  event_year
ORDER BY
  event_year ASC;

```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [event_year: integer, total_damage: double]





5. Repository de Scripts

Les scripts utilisés pour le projet sont disponibles sur https://github.com/moussatr/TP_Big_Data?tab=readme-ov-file#readme.

6. Conclusion

Ce projet a permis de créer un pipeline efficace pour l'analyse des impacts des catastrophes naturelles. Les données brutes ont été transformées en tables prêtes à l'emploi pour des visualisations dans Databricks.